ELECTRICAL
& COMPUTER
ENGINEERING

ECE-595 Network Softwarization

PROF. FABRIZIO GRANELLI (FABRIZIO.GRANELLI@UNITN.IT)

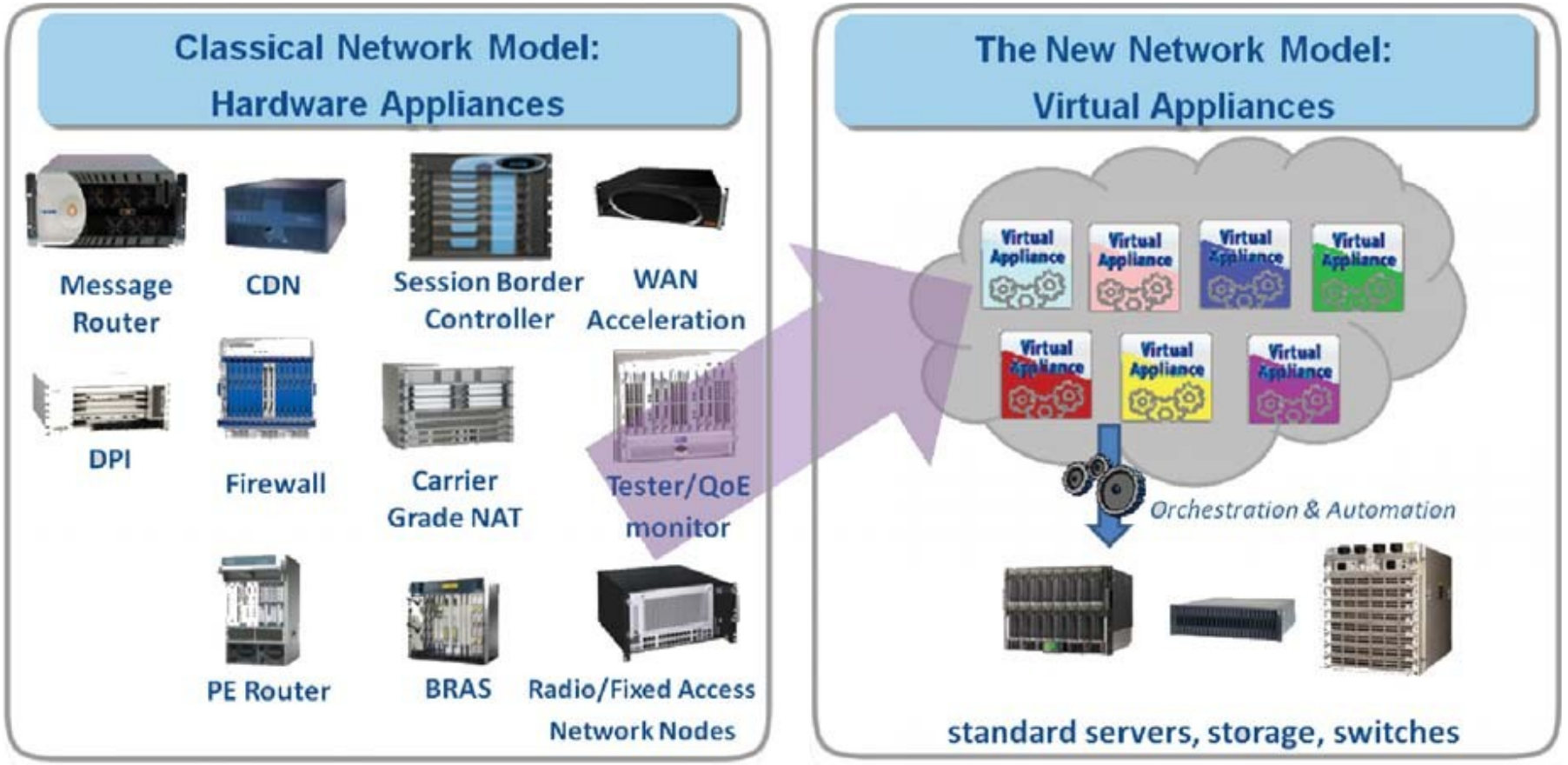
PROF. MICHAEL DEVETSIKIOTIS (MDEVETS@UNM.EDU)

Network Function Virtualization

*"NFV is a **network architecture concept** that proposes using IT virtualization related technologies to **virtualize** entire classes of **network node functions** into building blocks that may be connected, or chained, together **to create communication services**"*

Wikipedia:
http://en.wikipedia.org/wiki/Network_Functions_Virtualization

The NFV concept



NFV vs SDN

NFV (Network Function Virtualization) and SDN are complementary

- One does not depend upon the other.

Both have similar goals but approaches are very different

SDN needs new interfaces, control module applications.

NFV requires moving network applications from dedicated hardware to virtual containers on commercial-off-the-shelf (COTS) hardware

https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

NFV Components

Network Function (NF): Functional building block with well defined interfaces and well defined functional behavior

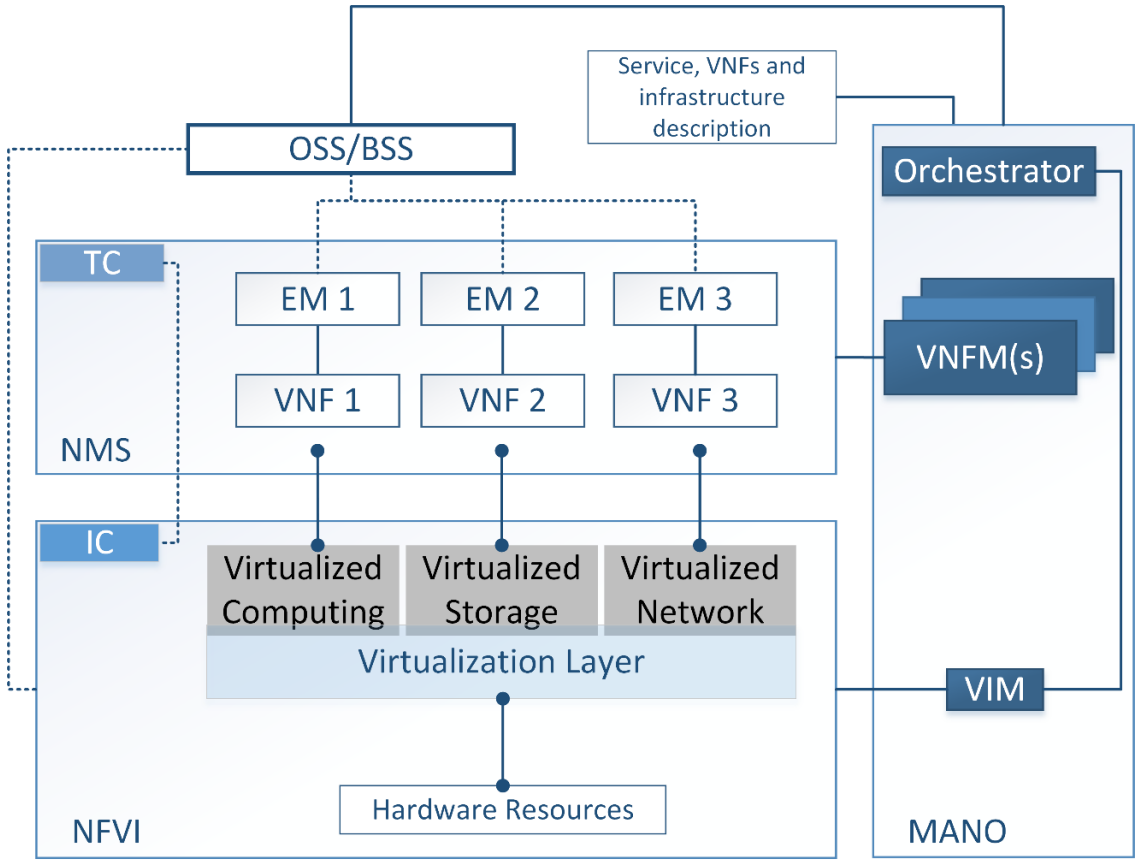
Virtualized Network Function (VNF): Software implementation of NF that can be deployed in a virtualized infrastructure

VNF Forwarding Graph: Service chain when network connectivity order is important, e.g. firewall, NAT, load balancer

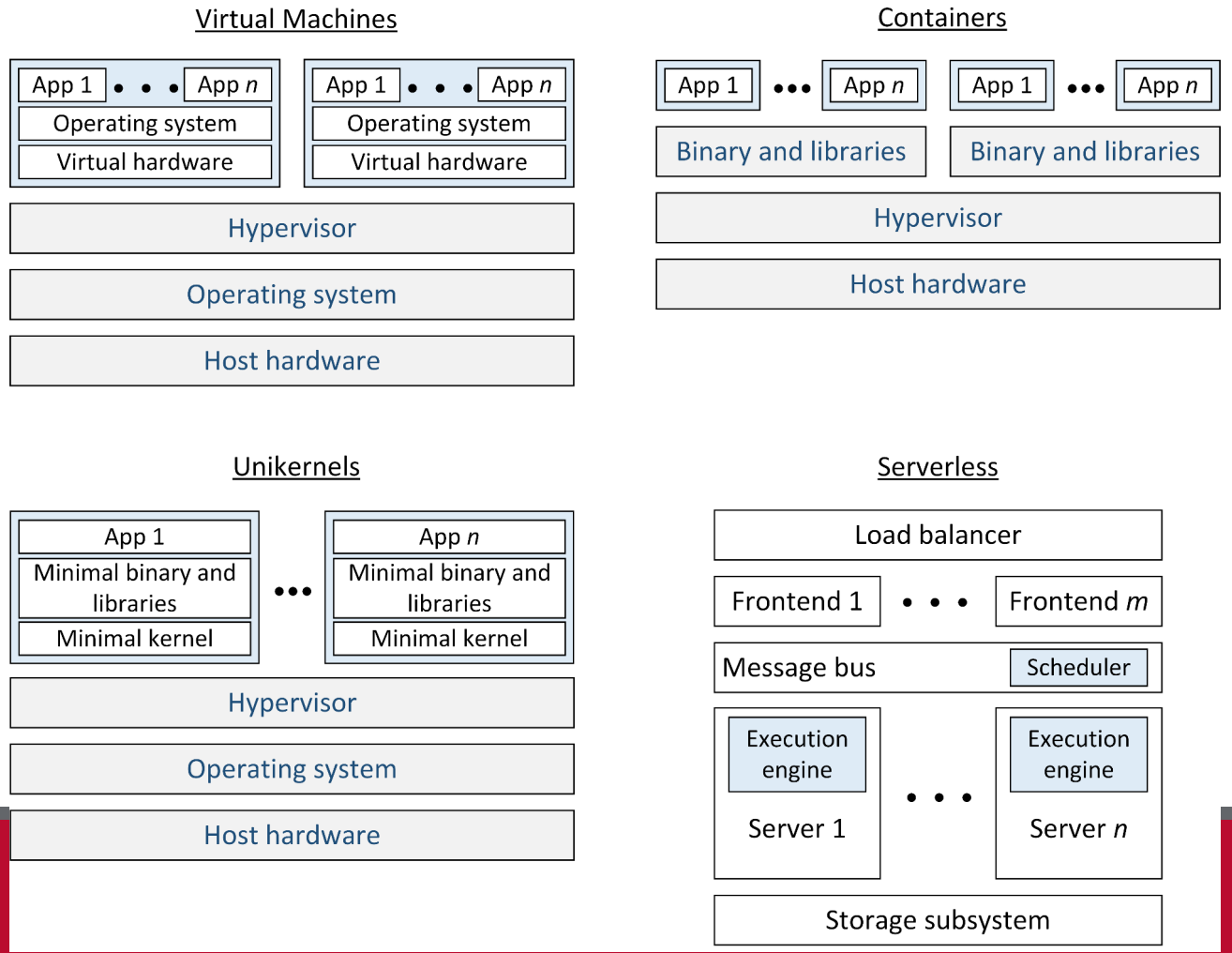
NFV Infrastructure (NFVI): Hardware and software required to deploy, manage and execute VNFs including computation, networking and storage

NFV Management & Orchestration: The orchestration of physical/software resources that support the infrastructure virtualisation, and the management of VNFs

ETSI SDN-NFV MANO architecture



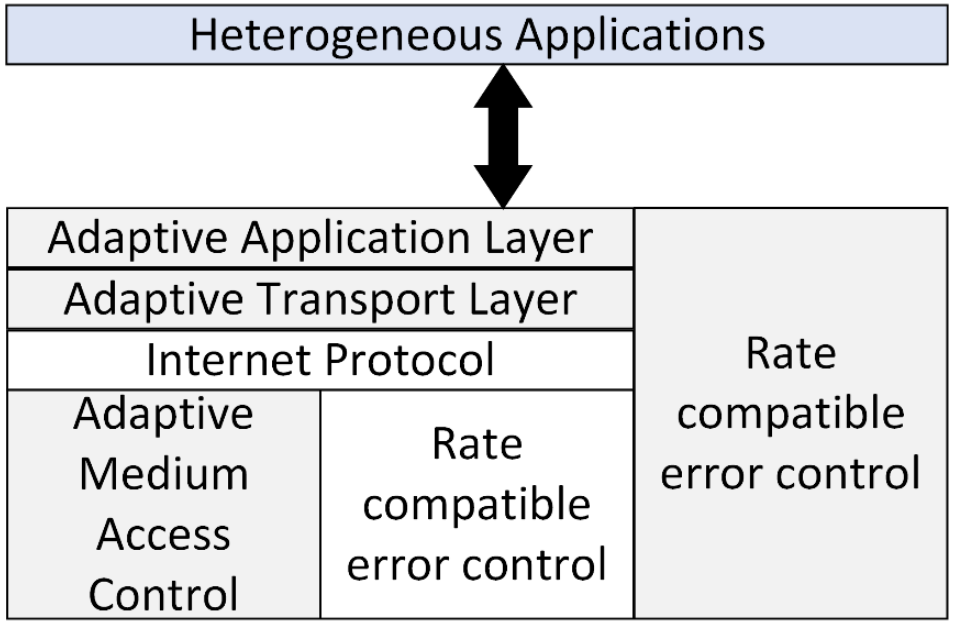
Architectures to realize network function virtualization



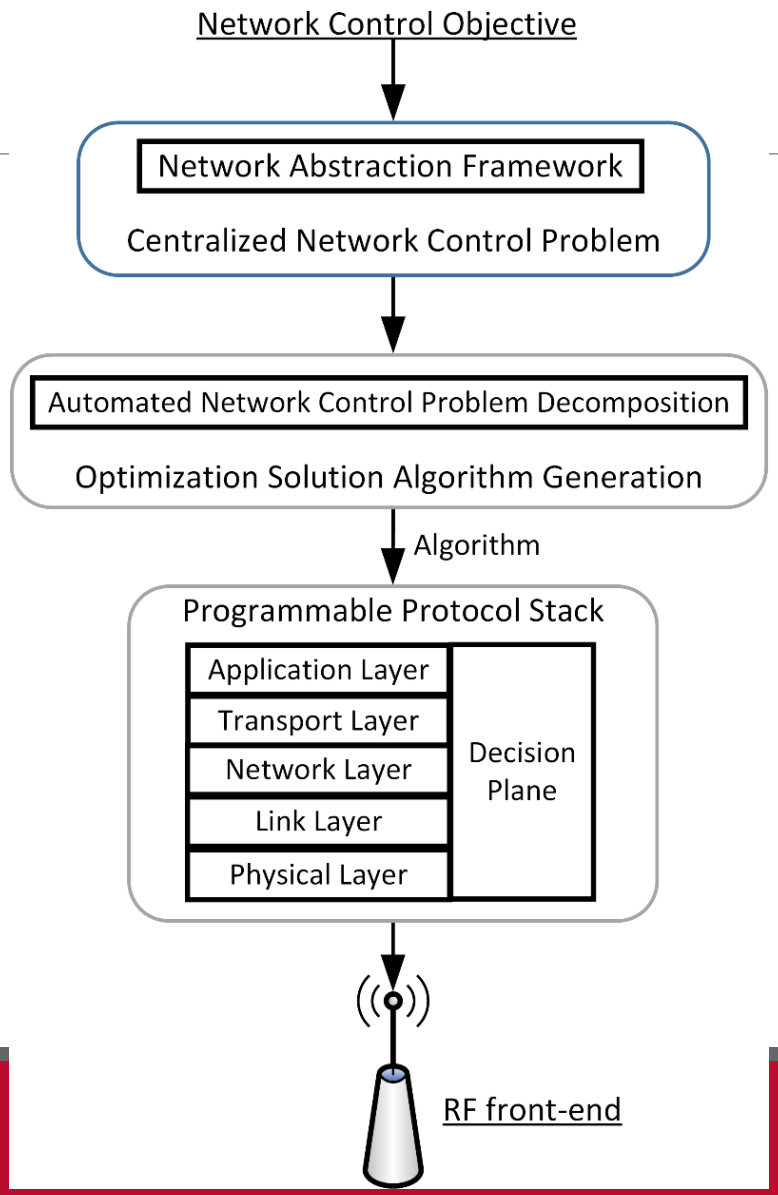
Programmable Protocol Stack

In depth penetration of the concept of NFV enables the protocol stack itself to become adaptable and programmable

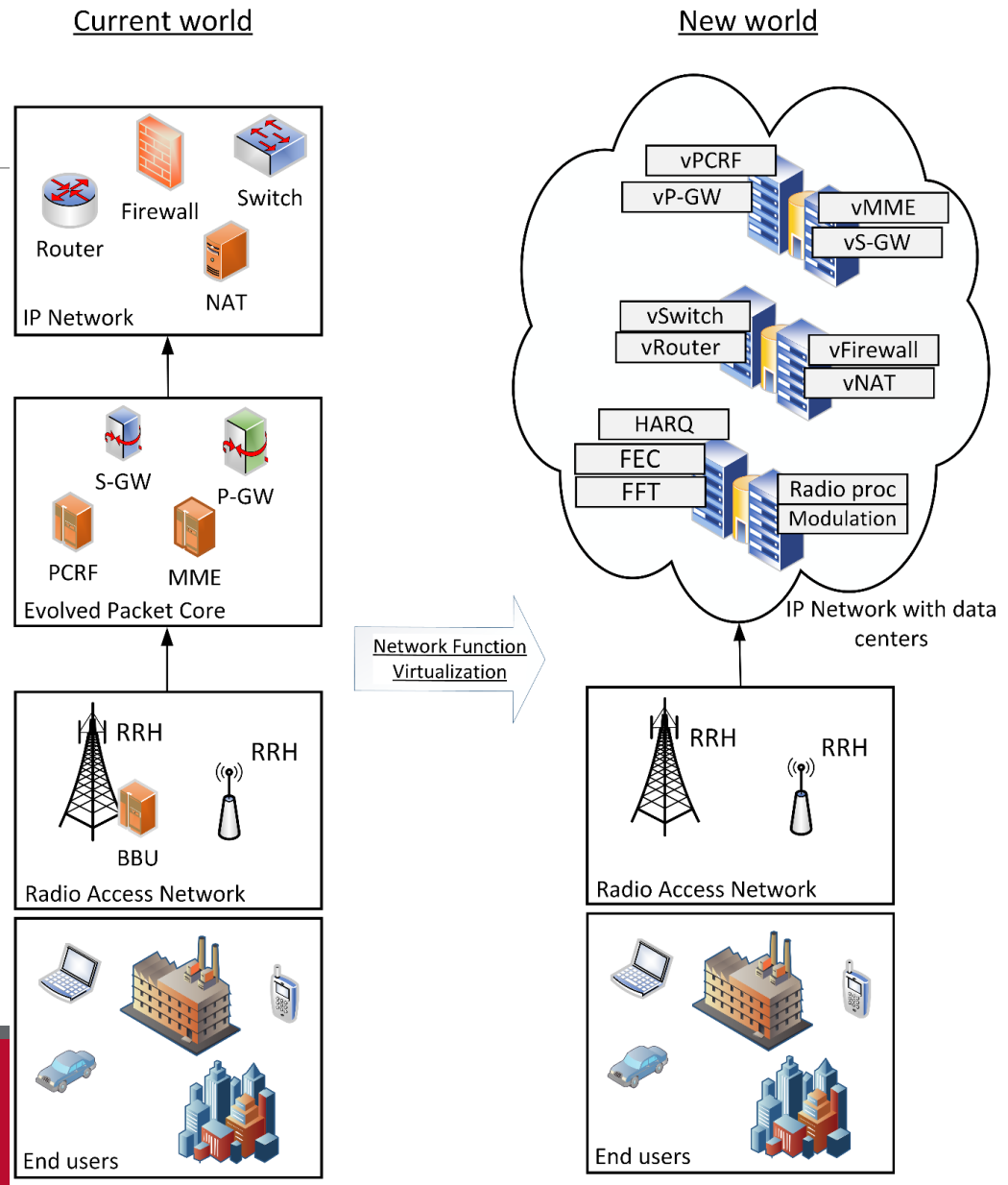
E.g. AdaptNet (adaptive protocol stack)



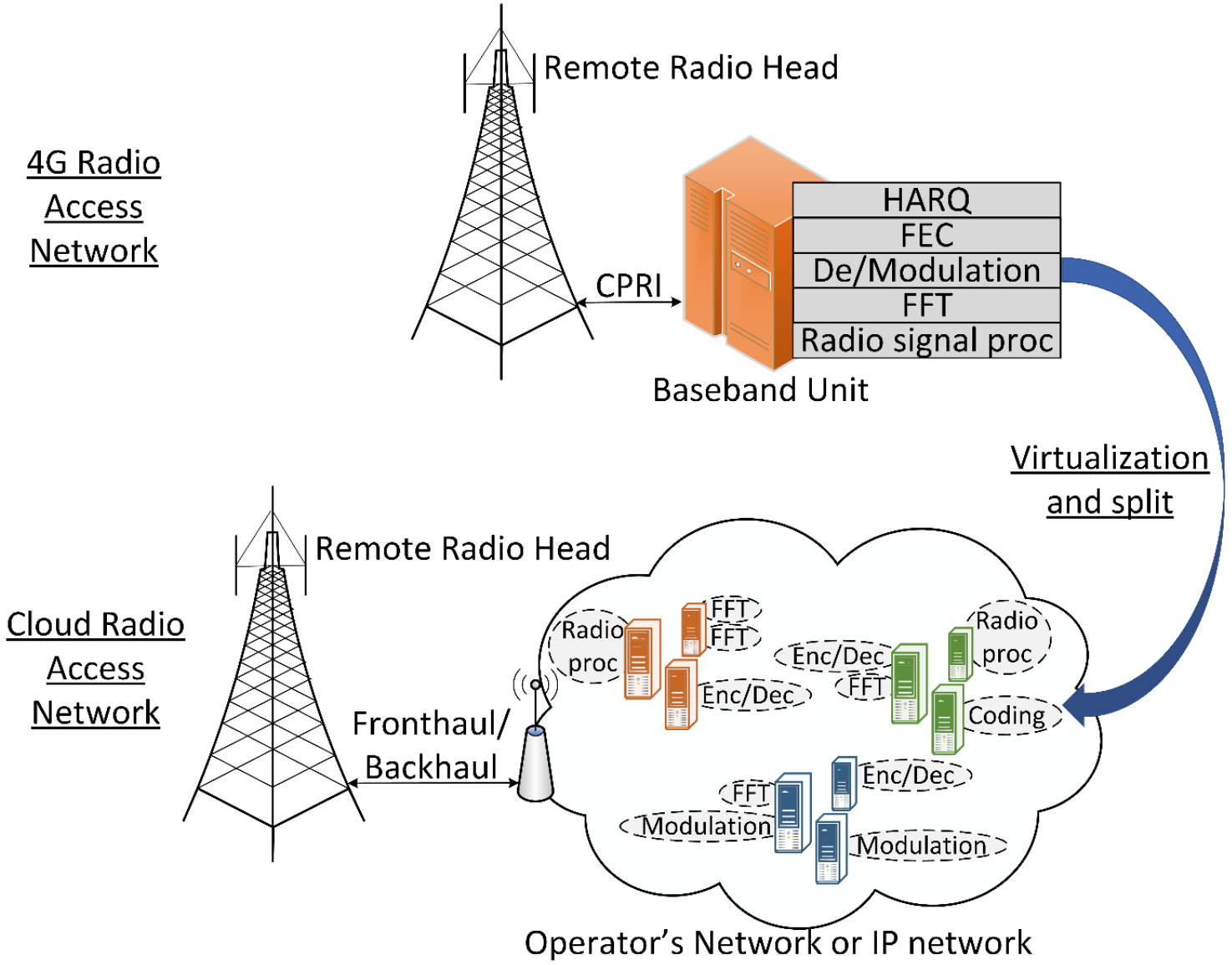
Architecture of wireless network operating system



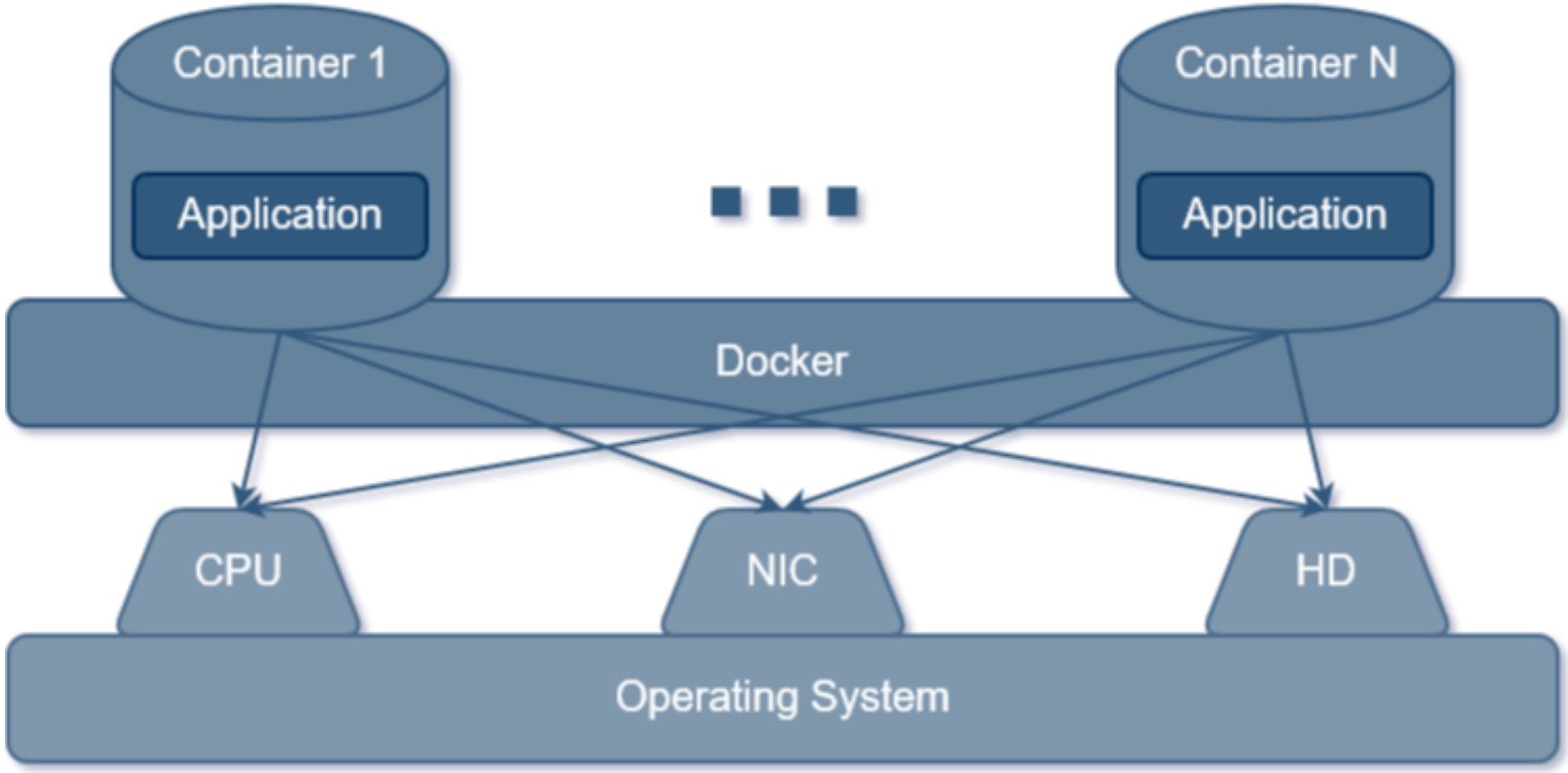
4G wireless cellular network: from legacy to virtualization of network functions



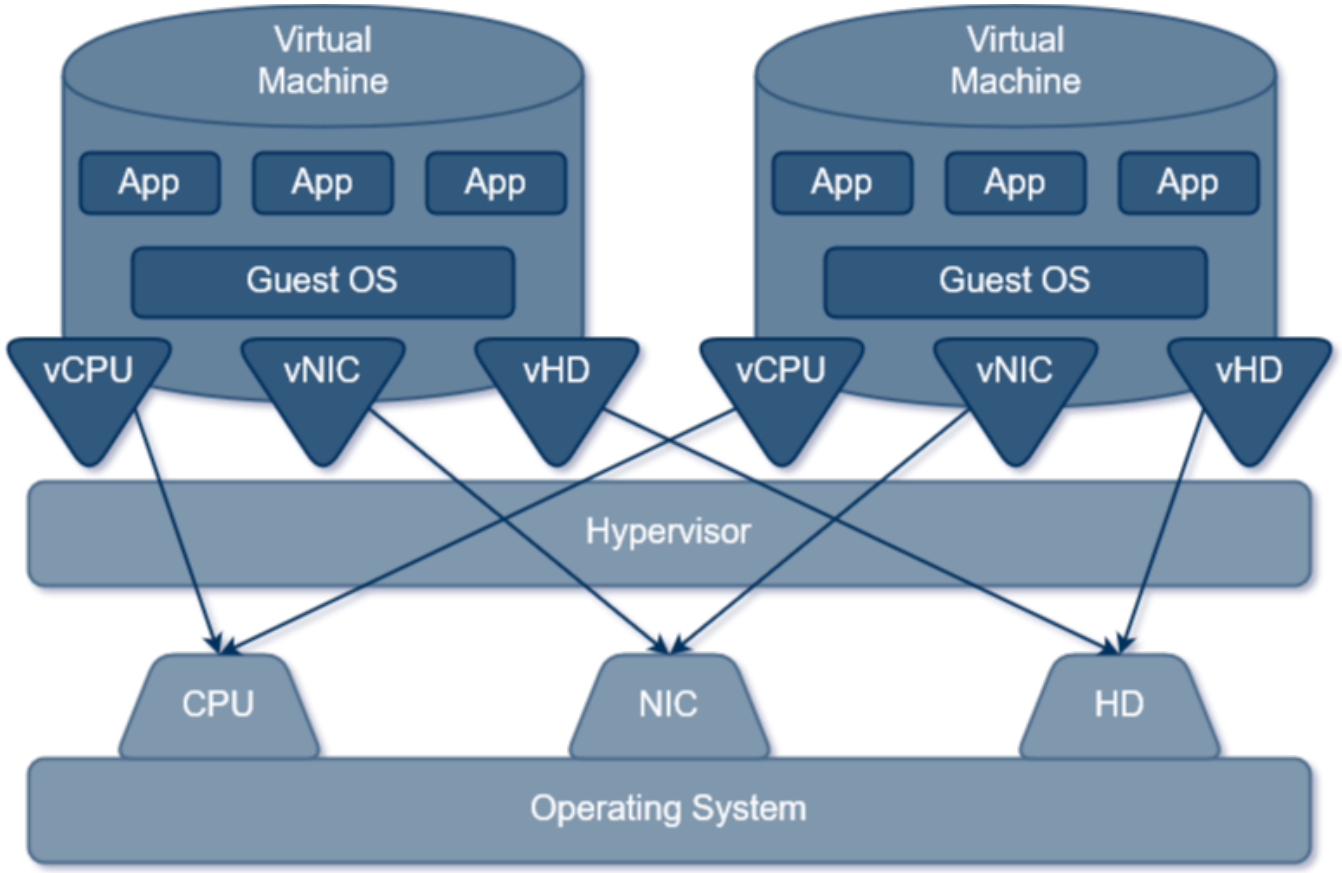
Virtualization of RAN and BBU Splitting



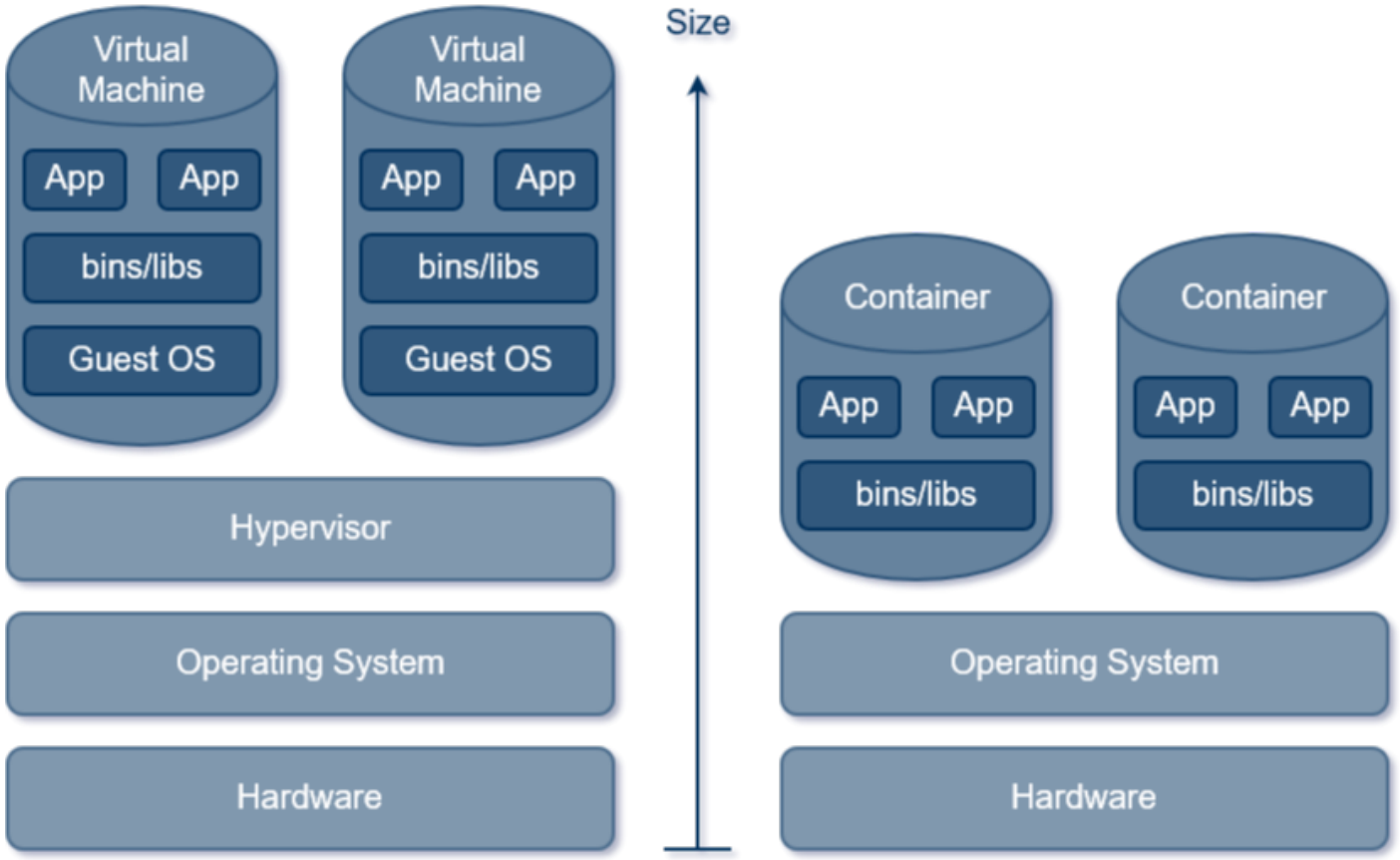
Docker: containerize your application



Virtualization with VMs



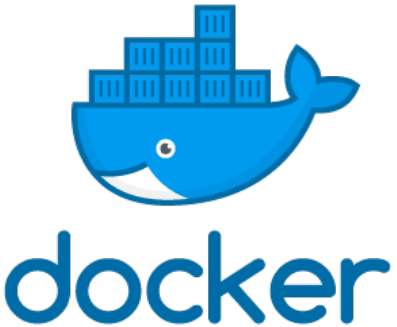
Size comparison between VMs and containers



Docker: containerize your app

Docker is already installed in the ComNetsEmu package

Docker is a simple and multi-platform solution for creating and managing containers



Docker basics

Check docker version:

```
$ docker -v
```

Running your first container (you might need to register to Docker Hub):

```
$ docker run docker/whalesay cowsay Hello World
```

How does it work?

Try:

```
$ docker run hello-world
```

Some more information

Check which images you have:

```
$ docker images
```

Download a new image (minimal Linux):

```
$ docker pull alpine
```

To delete an image:

```
$ docker rmi <image>
```

To check active containers / stop / delete:

```
$ docker ps
```

```
$ docker stop CONTAINER_ID
```

```
$ docker rm CONTAINER_ID
```

Running docker containers

Some ways to run docker containers:

```
$ docker run alpine ping "8.8.8.8"
```

```
$ docker run -t -i ubuntu /bin/bash
```

```
$ docker run -d ubuntu /bin/sh -c "while true; do $(echo date); sleep 2 ; done"
```

Docker containers are isolated, try INSIDE a container:

```
[container]$ rm -rf / --no-preserve-root
```

NEVER DO THIS IN YOUR OWN SYSTEM!

Set up a Web Server in seconds!

Let’s run nginx web server container:

```
$ docker run -d -p 80:80 nginx
```

Then, open your browser to localhost:80

Creating our own image

First, let's prepare an html file for our server:

FILE index.html

```
<h1> Welcome to ComSoc Training! </h1>

<h2> Have Fun! </h2>
```

Then, we need a Dockerfile to build our image:

FILE Dockerfile

```
# Base image
FROM nginx:latest
# The maintainer
MAINTAINER Fabrizio

# The actual recipe
# ADD Will add the created file in the step above
ADD ./index.html /usr/share/nginx/html/index.html
EXPOSE 80
```

Building and running the image

Now we are ready to build and run our image:

```
$ docker build -t my_server .
```

```
$ docker run -p 80:80 -d my_server
```

Other Useful Commands

List all containers (only IDs):

```
$ docker ps -aq
```

Stop all running containers:

```
$ docker stop $(docker ps -aq)
```

Remove all containers:

```
$ docker rm $(docker ps -aq)
```

Remove all images:

```
$ docker rmi $(docker images -q)
```

Advanced Usage

Deploy your own Minecraft server:

```
$ docker run -d -p 25565:25565 -e EULA=TRUE -e ONLINE_MODE=FALSE itzg/minecraft-server
```

Deploy your own repository:

```
$ docker run -d -p 5000:5000 --restart always --name registry registry
```

Save your container locally:

```
$ docker save busybox > busybox.tar
Or
$ docker save --output busybox.tar busybox
Or
$ docker save myimage:latest | gzip > myimage_latest.tar.gz
```